SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

A Method for Availability Monitoring Via A Shared Database

Background of Invention

[0001] The present invention relates to a method and means of indicating and determining the availability of a multitude of application servers providing application services to a multitude of application clients.

[0002] Enterprises depend on the availability of the systems supporting their day to day operation. A system is called available if it is up and running and is producing correct results. In a narrow sense availability of a system is the fraction of time it is available. MTBF denotes the mean time before failure of such a system, i.e. the average time a system is available before a failure occurs (this is the reliability of the system). MTTR denotes its mean time to repair, i.e. the average time it takes to repair the system after a failure (this is the downtime of the system because of the failure). Then, AVAIL=MTBF/(MTTR+MTBF) is the availability of the system. Ideally, the availability of a system is 1. Today, a system can claim high availability if its availability is about 99.999% (it is called fault tolerant if its availability is about 99.99%). J. Gray and A. Reuter, "Transaction processing: Concepts and Techniques", San Mateo, CA: Morgan Kaufmann 1993 give further details on these aspects. Availability of a certain system or application has at least two aspects: in a first, narrow significance it relates to the question, whether a certain system is active at all providing its services; in a second, wider significance it relates to the question, whether this service is provided in a timely fashion offering a sufficient responsiveness.

[0003] One fundamental mechanism to improve availability is based on "redundancy": The availability of hardware is improved by building clusters of machines and the

availability of software is improved by running the same software in multiple address spaces.

[0004] With the advent of distributed systems, techniques have been invented which use two or more address spaces on different machines running the same software to improve availability (often called activereplication). Further details on these aspects may be found in S. Mullender, "Distributed Systems", ACM Press, 1993. In using two or more address spaces on the same machine running the same software which gets its request from a shared input queue the technique of warm backups is generalized by the hotpool technique.

[0005] C. R. Gehr et al., "Dynamic Server Switching for Maximum Server Availability and Load Balancing", U.S. Pat. No. 5,828,847, which is hereby incorporated herein by reference, teaches a dynamic server switching system related to the narrow significance of availability as defined above. The dynamic server switching system maintains a static and predefined list (a kind of profile) in each client which identifies the primary server for that client and the preferred communication method as well as a hierarchy of successively secondary servers and communication method pairs. In the event that the client does not have requests served by the designated primary server or the designated communication method, the system traverses the list to obtain the identity of the first available alternate server-communication method pair. This system enables a client to redirect requests from an unresponsive server to a predefined alternate server. In this manner, the system provides a reactive server switching for service availability.

[0006] In spite of improvements of availability in the narrow sense defined above this teaching suffers from several shortcomings. Gehr's teaching provides a reactive response only in case a primary server could not be reached at all. There are no proactive elements which already prevent that a client requests service from a non-responsive server. As the list of primary and alternate servers is statically predefined there may be situations in which no server could be found at all or in which a server is found not before several non-responsive alternate servers have been tested. In a highly dynamic, worldwide operating network situation where clients and servers permanently enter or leave the network and where the access pattern to the servers

may change from one moment to the next, Gehr's teaching is not adequate.

[0007] The European Patent application EP 99109926.8 titled "Improved Availability in Clustered Application Servers" by the same inventors as the current invention is also related to the availability problem and any U.S. Patent based on this EP Application is hereby incorporated herein by reference. But this teaching is solely focused on the side of the application client. To make sure that a certain application request is being processed by an available application server it is suggested to send this application requests in a multi-casting step to a multitude of application servers in parallel assuming that at least one available application server will receive this request. This teaching is completely mute on techniques of how to indicate availability of a certain application server.

[0008] From the same inventors a further European Patent application EP 99122914.7 titled "Improving Availability and Scalability in Clustered Application Servers" is known and any U.S. Patent based on this EP Application is hereby incorporated herein by reference. In this application the existence of a technique to determine availability of an application server is already assumed as a starting point. This teaching is then focusing on a technique of how an application client can perform workload balancing by selecting a certain application server to process an application request.

[0009] Despite the progress thus far, further improvements are urgently required supporting enterprises in increasing the availability of their applications and allowing for instance for electronic business on a 7 (days) * 24 (hour) basis; due to the ubiquity of worldwide computer networks at any point in time somebody might have interest in accessing a certain application server.

[0010] The invention is based on the objective to provide an improved method and means for indicating availability of application servers to accept application requests and to provide an improved method and means for determining by an application client availability of an application server.

[0011] It is a further objective of the invention to increase the availability by providing a technology, which is highly responsive to dynamic changes of the availability of individual application servers within the network.

Summary of Invention

[0012] The objectives of the invention are solved by the independent claims. Further advantageous arrangements and embodiments of the invention are set forth in the respective subclaims.

[0013] The proposed method comprises for each of application server a first step of inserting into an availability database a notification period defining an upper time limit for a repetition period of an availability signal, which is repeated as long as the application server is available. In a second step for each availability signal its corresponding time stamp is inserted as availability time into the availability database. The difference of the current time and a recent availability time compared to said notification period is representing a measure of availability for the application servers.

Brief Description of Drawings

[0014] Figure 1 is a diagram reflecting the concepts of an application server, a hot pool, an application cluster and an application client.

[0015] Figure 2 reflects the suggested availability database according to the current invention, which is maintained by each application server/corresponding watchdog as a communication medium for indicating its availability status.

[0016] Figure 3 shows the record format of the period table according to the current invention comprising the individual notification periods.

[0017] Figure 4 visualizes the record format within the availability database to store the individual availability signals.

[0018] Figure 5 reflects a flow diagram depicting the method and computer program product for indicating availability according to the current invention also including the dynamic aspect of adapting the notification period depending on the workload situation.

[0019] Figure 6 shows an example of an implementation combining the period table and the availability signal table into a single table only.

Detailed Description

[0020] The proposed technology increases the availability and scalability of a multitude of application servers providing services to a multitude of application clients. The current invention is providing a proactive technology as it prevents that an application client generates erroneous request routings requesting service from non-responsive servers. The dynamic technique with ongoing processing is highly responsive to dynamic network situation where clients and servers permanently enter or leave the network. Thus the invention can accommodate hot plug-in of server machines into application clusters, thus further increasing the scalability of the environment. Complicated administration efforts to associate application clients with application servers are complete avoided.

[0021] The present invention can be realized in hardware, software, or a combination of hardware and software. Any kind of computer system - or other apparatus adapted for carrying out the methods described herein - is suited. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention can also be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which - when loaded in a computer system - is able to carry out these methods.

[0022] Computer program means or computer program in the present context mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following a) conversion to another language, code or notation; b) reproduction in a different material form.

[0023] If the current specification is referring to an application it may be a computer program of any nature not limited to any specific type or implementation. The terms application client and application server have to be understood from a logical point of view only relating to some type of "instance". These terms do not distinguish necessarily different address space or even different computer systems.

[0024] The current invention is assuming a certain communication path between application client and application server; this does not mean that the invention is

limited to a certain communication paradigm.

[0025] Also if the current specification is referring to a "database" the term is to be understood in a wide sense comprising not only actual databases (like relational, hierarchical databases etc.) but also simple files and the like. In other words the term database refers to any type of persistent storage.

[0026] Enterprises depend on the availability of the systems supporting their day to day operation. A system is called available if it is up and running and is producing correct results. In a narrow sense the availability of a system is the fraction of time it is available. In a second, wider sense availability relates to the question, whether an application service is provided in a timely fashion offering a sufficient responsiveness.

[0027] In the most preferable embodiment the current invention is relating to environments called "application cluster" based on the following concepts which are also depicted in Figure 1:

[0028] An application server (110, 111 or 112) is an executable implementing a collection of related services - for instance including access to some shared remote database (100). A hotpool (110, 111, 112) is a collection of address spaces each of which runs the same application server and each of these application servers receive requests from an input queue (125), which is shared between the hot pool members. By a servermachine (101, 102 or 103) we mean a certain physical machine which hosts a hot pool of application servers. An applicationcluster (120) is a collection of servers which fail independently and each server hosts a hot pool of application servers of the same kind.

[0029] Applications (130) request services from application servers via application clients. An applicationclient (131) is an executable which runs on the same machine as the application and which communicates with a server on behalf of the application. If the communication between the application client and a server is based on (asynchronous) reliable message exchange, the application server is said to be message based. In what follows we do assume message based communication between application clients and application servers; of course the invention is not limited to the message based communication paradigm as other paradigms may be

used instead. Consequently, an application client requests the performance of a certain service by sending a corresponding message into the input queue of a hot pool of associated application servers on a particular machine.

[0030] A client can protect itself against server failures and thus increasing the availability of the overall environment by simply multi-casting its requests as already described above in conjunction with the European Patent Application EP 99109926.8. But this requires a special implementation of the application servers or it is restricted to idempotent requests. Furthermore, it increases the number of messages sent by factors ! If the number of messages is a problem, each client that sends requests to a hot pool has to detect that this hot pool has failed (which is easy: the corresponding PUT command will be negatively acknowledged to the client by the messaging middleware!). When the client would know other hot pools of the same application server (i.e. server machines of the application cluster the failing hot pool is a member of) it could send its requests to another hot pool on a different server of the cluster. In doing so, clients could implement takeover between hot pools themselves.

[0031] Therefore the problem is to detect servers that are still available for accepting requests (so-called availability monitoring). For that purpose a so-called watchdog can be used to monitor a hot pool on a single machine to detect failed servers. In addition, a watchdog will automatically restart failed application servers of the hot pool it monitors. In conjunction with the European Patent Application EP 99122914.7 as described above the concept of watchdog monitoring has been discussed to detect failed server machines in application clusters. This concept is based on a specific communication protocols between the watchdogs to monitor and to determine the set of available application servers. Typically, messages are sent via the network between parts of a distributed application to maintain the overall state of its components. Considering the collection of watchdogs to be monitored as such a distributed application (the sole purpose of which would be to respond to inquiries about the liveliness of its distributed components) such a network-based message passing scheme can be used. But network-based message passing protocols have a couple of inherent problems (more or less severe), for example,

[0032]

1.the simple fact, that messages are to be sent will put additional load on the

network which is not tolerable in some situations;

[0033] 2.more complex algorithms have to be implemented to avoid single points of failures (like in "centralized" monitoring where a distinguished watchdog simply observes the others as participants) which results in a more development efforts; moreover such implementations raise the problem of "check the checker", i.e. specific programming techniques have to be exploited to make sure that these checking instances themselves do not create any failure.

[0034] 3.reachability properties must be ensured (e.g. the central watchdog must be able to reach all others in "centralized monitoring", or each watchdog must be able to reach all others in "distributed monitoring") which is both, a hard to achieve administrative task in setting up the environment appropriately, and difficult to cope with in case of network partitioning (i.e. due to connection losses the network dissociates into disjunct sub-nets) that can occur and must be handled.

[0035] As a consequence, the objective of the current invention is to overcome such mechanisms that require these extensive network based message passing protocols. But at the same time the desired solution to these problems should provide a proactive technology which automatically determines application servers entering (hot plug-in) or leaving the cluster.

[0036] The central idea of the current invention is reflected in Fig. 2. The central observation of the current invention is, that introduction of a central and shared database would reduce the network message traffic issue mentioned above significantly. It is suggested to use a database shared by all watchdogs to be monitored as the communication medium for exchanging state about liveliness of application servers. This new database is referred to as lifedatabase or availabilitydatabase 200. In the preferred embodiment of the invention periodically each watchdog 201 of the corresponding application servers of the cluster 202 writes an "I am alive!" 203 record into the life database; this record is to be understood as an availabilitysignal of the corresponding application servers to indicate their readiness to accept application service requests. The introduction of the watchdog concept is already an additional improvement; of course it would be possible that each application server itself is responsible to insert the availability signal into the

availability database.

[0037] As a sample embodiment a relational database system hosting the live database of an application cluster is assumed. Note, that this is not central to the current invention, i.e. any other persistent store (e.g. a file system or an enterprise Java beans entity container) can be used for this. Especially, the topology database that an application cluster might use for its systems management could be extended via appropriate tables representing the life database.

[0038] Any software (for instance application clients interested in requesting application services) that can access the life database of an application cluster can determine the servers that are available and the ones that have failed and are currently not available.

[0039] It is not sufficient, that for a certain application server or its watchdog a corresponding availability signal would be entered into the availability database just once. If after this event the application server would crash, the availability database would run out of sync with the current situation. To cope with this problem the current invention suggests for this purpose, that the life database must also contain information about the period each watchdog agreed to write "I am alive!" records into the database. Therefore a further data element is to be inserted into the availability database comprising a notification period; the notificationperiod defines an upper time limit during which the availability signal is repeated as long as the corresponding watchdog (or application server) is available.

[0040] As a sample embodiment Fig. 3 shows the periodtable to store the individual notification periods. It is suggested that the period table comprises an identification of the watchdog (representing the application cluster) or an identification of the application server 300 which repeats the availability signal as well as the notification period 301. Each watchdog/application server participating in the availability monitoring would enter such an record into the availability database. It is obvious to every average expert of this technical field how to derive via SQL the period with that a watchdog writes "I am alive!" messages from that table. Also, all watchdogs encompassed by the application cluster can be derived from that table via SQL.

[0041] As a sample embodiment, Fig. 4 depicts the table Alive_Signal that is used to

represent an "I am alive!"record of a watchdog; that is, for each availability signal received from a watchdog such a record would be entered into the availability database. Similar to the period table, it is suggested that the Alive_Signal table comprises identifications 400 of the corresponding watchdog/application server which sent the availability signal. Moreover the Alive_Timestamp field 401 stores the time stamp and therefore the availability time of the most recent availability signal.

[0042] The information contained in these two tables, the period table and the Alive_Signal table, reflect a precise picture of the availability of the application servers.

[0043] In general terms for each application server an availability measure is defined by the difference of the current time (for instance the time when querying the availability database) and the most recent availability time in comparison with the notification period of that particular application server. Even more generally a second difference between the current availability time and the previous availability time can be added to the availability measure. The following specific availability measures have been proven to be successful:

[0044] 1.If the difference of the current time and the most recent availability time exceeds the notification period, the corresponding application server is treated as unavailable; this is because the application server "promised" to repeat availability signals at least within the notification period. Otherwise the application server is regarded as available.

[0045] 2.From the Alive_Signal table, the time passed between the insertions of the last two "I am alive!"records written by a particular watchdog can be determined; i.e. the time difference between the most recent availability time and the previous availability time is determined. If that duration exceeds the notification period this watchdog agreed to insert "I am alive!"messages, the watchdog is a candidate to have failed. This availability measure is based on the assumption that, if the last two availability signals are not within the expected notification period, this is a indication that the application server currently is experiencing problems and therefore it should be avoided.

[0046] 3.Typically, such a timeout based failure determination mechanism must cope

with situations like that a watchdog is simply too busy to write an availability signal into the availability database but is still available. An availability measure being able to cope with such a situation is achieved by treating an application server as unavailable, if the difference between the current time and the previous availability time exceeds the notification period by a factor of N.

[0047] However, based on the availability database (life database) it can be determined which watchdog/application server has failed and which watchdog/application server is still available. Especially, every program having access to the life database can perform this check: each watchdog, a separate administration component that might be build for this environment and of course each application client looking for an available application server for passing over an application service request. Each application client can query the availability database making use of above availability measures to determine at least one available application server to which it then sends an application service request.

[0048] A further advantageous embodiment of the current invention can be achieved if the watchdogs or application servers dynamically adjust their notification period. If this dynamic adjustment depends on the amount of workload to be processed by an application server, the availability measure becomes a new quality by also representing a workload indication. By increasing the notification period, if the amount of workload increases, and by decreasing the notification period, if the amount of workload decreases, the notification period represents (at the same time) an workload indicator expressing the responsiveness of an application server. This indication can be exploited by an application client by determining the availability measure for a set of application servers in parallel. In this situation the availability measure can be used not only for determining the subset of available application servers (which would represent a binary decision only: "available"versus "unavailable") but it also can form a basis for workload balancing decision executed by the application client. The numerical value of the availability measure, more or less influenced by the current notification period as a parameter, is then also a workload indication. An application client would then issue its application service request to that available application server with the lowest workload, i.e. the application server with the largest availability measure with respect to this further application request.

[0049] Fig. 5 reflects a flow diagram depicting the method for indicating availability also including the dynamic aspect of adapting the notification period depending on the workload situation. The process of availability monitoring by an application server or watchdog is started within step 501. Within the next step 502 the current workload situation is determined to calculate the notification period, which is not too high or too low compared to the current workload situation. This calculated notification period has to be entered (of course) into the availability database within step 503. Within the time frame set by the notification period the current availability signal has to be entered into the availability database; this is reflected in step 504. The notification period defines an upper time limit for repetition of the availability signal; depending on the workload the application server/watchdog will try to issue and availability signal more frequently. After step 504 (or as an alternative embodiment before this step) the current workload situation is analyzed within step 505. If the current workload situation changed in a way, which requires to re-adjust the notification period, the process step of determining the notification period is started again choosing the control path 506. If the current workload situation didn't change significantly, repetition of issuing an availability signal is repeated choosing path 507.

[0050] The structure and layout of the availability database with its period table (depicted in Fig. 3), it's availability table (depicted in Fig. 4) has to be understood from an expanded perspective only. Of course the structure of the availability database may be subject of further improvements like the following:

[0051] 1.Each insertion of a new notification period, or a new availability signal would introduce a new record into the database. To prevent that the availability database would permanently increase, processes are suggested, which remove old database records, which are of no use any more; for instance, with respect to each record type of a certain watchdog/application server only the most current and the previous record are maintained within the database. For the implementation of such a process the technology of "Stored Procedures" could be exploited advantageously; such an adapted stored procedure could run in the database in the background to delete records no longer required.

[0052] 2.It is of course not essential to the current invention to store the notification

period and the availability signal in different database records. An example how to include both data elements within one database record is visualized in Fig. 6. As can be seen from Fig. 6, besides the watchdog identification/application server identification 600 and the notification period 601, the multitude of availability signals is reduced to two entries only. Whenever the current availability signal 602 is updated by a new availability signal its contents is transferred into the field storing the previous availability signal 603; after that, the new availability signal is inserted into the field of the current availability signal 602. With this technique the availability database is limited to a moderate size as for each watchdog/application server a single database record has to be maintained only.

[0053] The proposed technology increases the availability and scalability of a multitude of application servers providing services to a multitude of application clients. The current invention is providing a proactive technology as it prevents that a client generates erroneous request routings requesting service from non-responsive servers. An ongoing process is suggested being highly responsive to dynamic network situation where clients and servers permanently enter or leave the network. Thus the invention can accommodate hot plug-in of server machines into application clusters further increasing the scalability of the environment. Complicated or due its sheer complexity impossible administration efforts to associate application clients with application servers are complete avoided.

[0054] As the current invention does not assume any network-based message passing, all disadvantages of such mechanisms (refer to the remarks above) are avoided. The only system prerequisite is a shared database. Today's database management systems are extremely robust such that one doesn"t have to consider the life database as a single point of failure. Furthermore, most application servers are built on top of a database system. Thus, the assumption of a shared database is automatically fulfilled in many situations. Reachability is not a problem at all because each server machine by hosting a hot pool has access to the shared database. Finally, the watchdog monitoring technique can be easily implemented via SQL when putting the life database into a relational DBMS.

[0055] It is to be understood that the provided illustrative examples are by no means

exhaustive of the many possible uses for my invention.

[0056] From the foregoing description, one skilled in the art can easily ascertain the essential characteristics of this invention and, without departing from the spirit and scope thereof, can make various changes and modifications of the invention to adapt it to various usages and conditions.

[0057] It is to be understood that the present invention is not limited to the sole embodiment described above, but encompasses any and all embodiments within the scope of the following claims: